МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

Тема: Бинарные деревья

Студент гр. 8304 _____ Ивченко А.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2019

Цель работы.

Ознакомиться со структурой бинарных деревьев; изучить варианты обхода бинарного дерева.

Задание(вариант 9-д).

Рассматриваются бинарные деревья с элементами типа Elem (в качестве Elem использовать char). Заданы перечисления узлов некоторого дерева b в порядке ЛКП и ЛПК.

Требуется:

-восстановить дерево b и вывести его изображение;

- перечислить узлы дерева b в порядке КЛП.

Описание работы алгоритма.

Была реализована струтура Node, имеющая поля: элемент узла и два указателя, которые по умолчанию равны nullptr.

```
struct Node {
char root;
Node* lst;
Node* rst;
Node() {
    lst = NULL;
    rst = NULL;
}
}
```

Были определены следующие функции:

typedef Node bTree*

bTree Create(void) //функция создания пустого бинарного дерева

bool isNull(bTree) //функция проверяющая бинарное дерево на пустоту

char RootBT(bTree) // функция, возвращающая корень бинарного дерева

bTree Left(bTree) //функция, возращающая указатель на левое поддерево

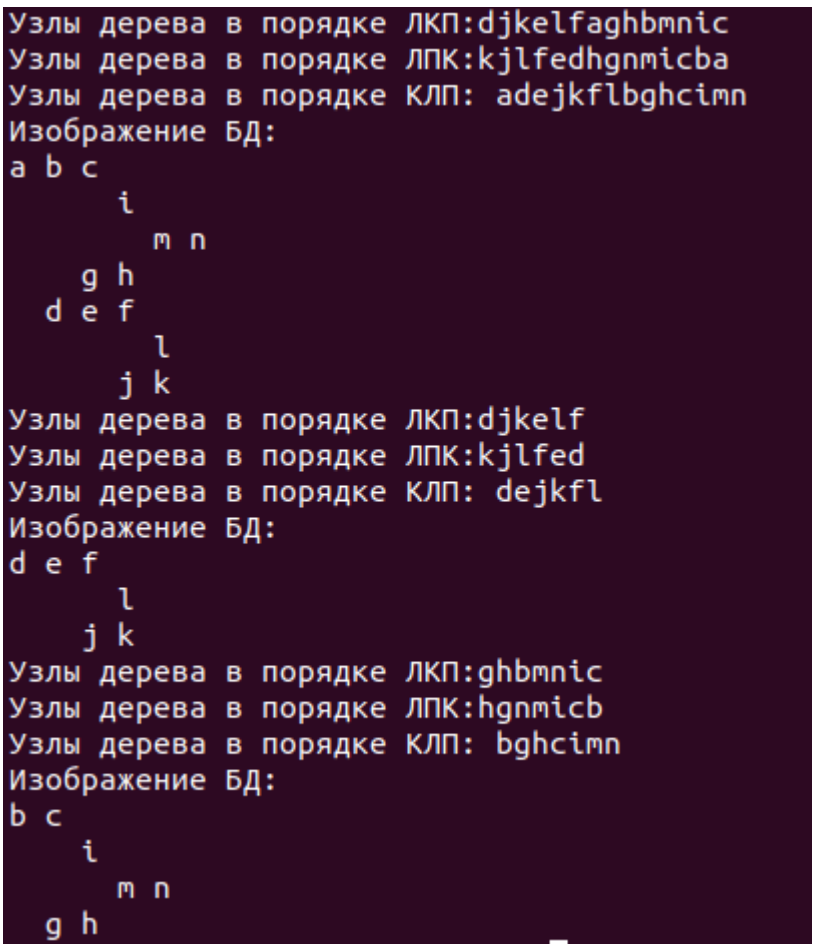
bTree Right(bTree) //функция, возращающая указатель на правое поддерево

bTree consBT(const char& x, bTree& lst, bTree& rst) //конструктор БД

void destroy(bTree&) //функция высвобождения памяти

void createTree(std::string lkp, std::string lpk, char node, std::vector<char>& klp, bTree &BinTree) //функция рекурсивно обрабатывает последовательности узлов в порядке ЛКП и ЛПК и формирует бинарное дерево в зависимости от условий по следующему алгоритму: при каждом вызове рекурсии в строке ЛПК удаляется последний символ, а в строке ЛКП удаляется символ, соответствующий удаленному так, чтобы в результате получилось две строки ЛКП (до удаленного элемента и после). ЛПК также разделяется по такому же соотношению. Получившиеся строки послужат аргументами к очередному вызову этой рекурсивной функции.

Тестирование.



Вывод.

В результате лабораторной работы была написана программа, восстанавливающая бинарное дерево из имеющихся во входных данных перечислений узлов в порядке ЛКП и ЛПК. Был получен опыт по представлению бинарного дерева в памяти, также были реализованы необходимые методы для работы с ним.

ИСХОДНЫЙ КОД

```
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
#include <vector>
#include "btree.h"


void createTree(std::string lkp, std::string lpk, char node,
std::vector<char>& klp, bTree &BinTree)
{
	bTree leftSubTree = Create();
	bTree rightSubTree = Create();

	klp.push_back(node);
	if (lpk.length() < 2){
		BinTree = new Node;
		BinTree->root = node;
		return;
	}

	int index = lkp.find(node);


	std::string leftlkp = lkp.substr(0, index);
	std::string leftlpk = lpk.substr(0, index);
	std::string rightlkp = lkp.substr(index + 1);
	lpk = lpk.erase(lpk.length()-1, 1);
	std::string rightlpk = lpk.substr(index);



	if (index!= 0){

		createTree(leftlkp, leftlpk, leftlpk[leftlpk.size() - 1], klp,
leftSubTree);
	}


	createTree(rightlkp, rightlpk, rightlpk[rightlpk.size() - 1], klp,
rightSubTree);

	BinTree = consBT(node, leftSubTree, rightSubTree);


	return;
}
void readFromFile(std::ifstream &file){
```

```
            int count;
            std::string lkp;
            std::string lpk;

            std::vector<char> klp;


            if(!file.is_open()){
                std::cout<<"Неверный путь к файлу\n";
                return;
            }
            std::string str;
            while (std::getline(file,str)){
                if (str[0] == ' ') continue;
                else{
                    if(count == 0) {lkp = str; count++;continue;}
                    else if(count == 1){lpk = str; count++; continue;}
                    else{
                    count = 0;

                    bTree MyBinTree;
                    createTree(lkp, lpk, lpk[lpk.size() - 1], klp, MyBinTree);


                    std::cout << " Узлы дерева в порядке ЛКП:" << lkp <<
std::endl;
                    std::cout << " Узлы дерева в порядке ЛПК:" << lpk <<
std::endl;
                    std::cout << " Узлы дерева в порядке КЛП: ";

                    for ( auto i: klp)
                        std::cout << i;

                    std::cout << std::endl;
                    klp.clear();
                    lkp.clear();
                    lpk.clear();

                    std::cout << " Изображение БД:" << std::endl;
                    displayBT (MyBinTree,1);
                    destroy(MyBinTree);
                    }
                }
            }
}

int main(int argc, char* argv[]){


    if (argc == 2) {
```

```
        std::ifstream file(argv[1]);
        readFromFile(file);

    }else{

    std::string lkp;
    std::string lpk;

    std::cout << " Введите последовательность узлов в порядке ЛКП:";
    getline(std::cin, lkp);
    std::cout << "\n Введите последовательность узлов в порядке ЛПК:";
    getline(std::cin, lpk);

    std::vector<char> klp;

    bTree MyBinTree;

    createTree(lkp, lpk, lpk[lpk.size() - 1], klp, MyBinTree);

    std::cout << " Узлы дерева в порядке КЛП:" << std::endl;

    for ( auto i: klp)
        std::cout << ' ' << i;
    std::cout << std::endl;

    std::cout << " Изображение БД:" << std::endl;
    displayBT (MyBinTree,1);
    destroy(MyBinTree);

    }return 0;
}
```